Cooperation of combinatorial solvers for en-route conflict resolution

Ruixin Wang, Richard Alligier, Cyril Allignol, Nicolas Barnier, Nicolas Durand, Alexandre Gondran

Cooperation of Combinatorial Solvers for En-Route Conflict Resolution

Ruixin Wang[a,b,*], Richard Alligier[b], Cyril Allignol[b], Nicolas Barnier[b], Nicolas Durand[b], Alexandre Gondran[b]

[a] *CAUC-ENAC Joint Research Center of Applied Mathematics for Air Traffic Management, Sino-European Institute of Aviation Engineering, Civil Aviation University of China, Tianjin, China*
[b] *ENAC, Université de Toulouse, France*

Abstract

One of the key challenges towards more automation in Air Traffic Control is the resolution of en-route conflicts. In this article we present a generic framework for the conflict resolution problem that clearly separates the trajectory and conflict models from the resolution. It is able to handle any kind of maneuver and detection models, though we propose our own realistic 3D maneuvers and conflict detection that takes into account uncertainties on the positioning of aircraft. Based on these models, realistic scenarios are built, for which potential conflicts are detected using an efficient GPU-based algorithm. The resulting instances of the conflict resolution problem are provided to the community as a public benchmark.

To efficiently solve this problem, we also introduce a generic framework for the cooperation of optimization algorithms. The framework benefits from the various optimization algorithms plugged to it by sharing relevant information among them, and is implemented as a distributed application for better performance. We illustrate its behavior on the conflict resolution problem with the cooperation between a Memetic Algorithm and an Integer Linear Program which consistently outperforms previous approaches by orders of magnitude. Instances with up to 60 aircraft are optimally solved within a few minutes using this framework, while each algorithm taken individually only provides sub-optimal solutions. This cooperative approach thus seems appropriate for application in a real-time context.

Keywords: conflict resolution, metaheuristics, integer linear programming, algorithm cooperation

[*] Principal corresponding author
Email addresses: ruixin.wang@recherche.enac.fr (Ruixin Wang), richard.alligier@enac.fr (Richard Alligier), cyril.allignol@enac.fr (Cyril Allignol), nicolas.barnier@enac.fr (Nicolas Barnier), nicolas.durand@enac.fr (Nicolas Durand), alexandre.gondran@enac.fr (Alexandre Gondran)

1. Introduction

The emergence of new ATM concepts such as free-routing or sector-less control will significantly modify the structure of en-route traffic in the future. This modification and the increase of traffic will directly impact controllers in their conflict management task. To tackle these rising issues, decision support tools for conflict detection and resolution are needed so as to alleviate the controllers workload and thus make it possible to accommodate for traffic increase.

Most of the mathematical models that have been developed to this purpose are very closely linked to the algorithmic method used for resolution, making any comparison of different approaches hardly objective. For example [13, 19] used the Base of Aircraft Data (BADA) developed and maintained by EUROCONTROL in the Complete Air Traffic Simulator (CATS) to solve conflicts using Evolutionary Algorithms. The solver was quite powerful as it could handle complete days of traffic in the European sky but the method was difficult to compare to others because the conflict detection was embedded in the solver. This problem also occurred in Erzberger's approach [16], where most of the expertise was focused on the trajectory and maneuver model. But results were impossible to compare to other methods because, once again, the resolution maneuver generator was embedded in the solver. In this article, we propose a framework for the en-route conflict resolution problem that strictly separates the trajectory and conflict modeling from the resolution, and provide a set of realistic instances as a freely and publicly available benchmark.

The modeling part is particularly versatile. It enables the representation of any discrete set of maneuvers, and is able to take into account uncertainties on many parameters of the trajectory. For the provided benchmark, we present in this article three kinds of maneuvers, namely a heading change, a Flight Level change or a speed change, which are commonly used by traffic controllers. For each kind of maneuver and for each aircraft, many trajectories are pre-computed (for example, a heading change can be parameterized by the angle of deviation). Potential conflicts among maneuvers are then detected using an efficient GPU-based algorithm to generate the conflict matrix of the resolution problem. For feasible instances, the best solution w.r.t. operational costs should be provided, considering airlines cost as well as Air Traffic Management cost. Therefore, our instances also specify a template[1] cost for each trajectory, favouring later, shorter and least-deviating maneuvers, and solvers should return (and prove when possible) optimal solutions w.r.t. the sum of the costs of assigned trajectories. The resulting public benchmark contains instances of various sizes, up to 60 aircraft with more than 190 possible trajectories per aircraft.

Given an instance, any algorithmic strategy can be used to solve the conflicts. In this article, we propose to use two classic algorithms tailored to improve their performance: a Memetic Algorithm, which is a metaheuristic based on a Genetic Algorithm and a Tabu Search, and an "aggregated" compact Integer Linear Programming method. We also propose a cooperative approach that benefits from the strength of both previous

[1] The costs defined in our model are arbitrary, though they allow to rank trajectories relatively according to their "efficiency". However, any more concrete cost model can be used with our approach.

methods and consistently increases the efficiency of the resolution tool by orders of magnitude over previously published results.

The paper is organized as follows. Previous research works related to the conflict resolution problem are presented in Section 2. Section 3 describes a maneuver model allowing heading, Flight Level or speed changes and a 3D-trajectory model that takes into account realistic uncertainties to detect all potential conflict. Section 4 details a Metaheuristic approach and a Integer Linear Programming method, as well as a framework for their cooperation. Section 5 compares both optimization techniques and their cooperation on our benchmark in order to assess the performance of these different approaches, showing the advantage of the cooperation method for large instances. We then conclude and present future research directions in Section 6.

2. Related Works

Research on automated aircraft conflict resolution started in the 1980s. Many different models were introduced to comply with existing resolution techniques. Some studies like [16], conducted by Air Navigation System Providers, offer realistic models but do not focus on the resolution methods. Other approaches, like [13, 19] which use uncertainty models and the Base of Aircraft Data (BADA, developed and maintained by Eurocontrol), studied both the model and the resolution algorithms. However, they were completely tailored to the underlying traffic simulator (CATS), which prevents the scientific community from comparing different resolution methods.

Many mathematical models have led to specific resolution algorithms able to deal with very complex situations, but require specific characteristics for trajectory prediction. This is the case for Pallottino's approach [29] that used Mixed Integer Linear Programming (as [35, 5, 32]). Theses models rely on constant speed trajectories and assume that all maneuvers are executed simultaneously. They cannot deal with trajectory models able to handle descending or climbing aircraft, nor with complex trajectory uncertainties. Other authors like [6, 7] proposed different Mixed Integer Non-Linear models that can deal with horizontal and vertical maneuvers, taking multi-objective criteria into account, though uncertainties are not included in the trajectory prediction.

Conflict resolution is known for being highly combinatorial [14] and large instances can therefore be very difficult to solve. Assessing the relative merits of different solvers is very useful to pave the way to future automation tools. Moreover, the best solution or at least a good solution is needed very quickly for real-time en-route conflict resolution. The computation time is thus critical.

In 2013, we proposed a framework to separate the trajectory model from the resolution algorithm [3], as well as a Constraint Programming (CP) approach to solve the problem with 2D maneuvers only. This framework was extended in 2017 to scenarios involving several Flight Level with 3D maneuvers in [4], and a second approach using a Memetic Algorithm (MA) was proposed. In these studies however, optimal solutions could only be found on small instances. On larger instances, the MA was able to find good solutions without reaching the optimal, while CP was occasionally able to prove optimality or unfeasability for highly constrained instances. The strict separation between the model and the resolution made it possible for us to publish the instances of the problem for the scientific and ATM community. This offered the

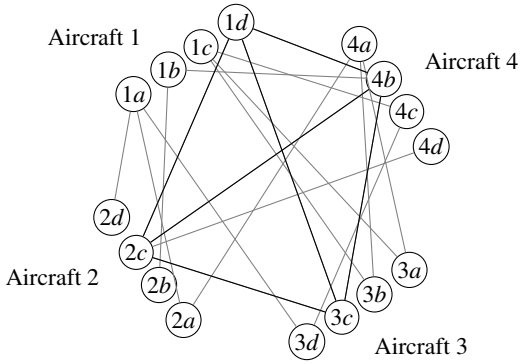


Figure 1: Graph representation of a conflict resolution problem: clique $\langle 1d, 2c, 3c, 4b \rangle$ is a solution.

opportunity to test different algorithms on various problems without investing effort in the model. With such a framework, resolution times and costs of different solvers can be fairly compared. Instances of the problem can be downloaded on the dedicated website: `clusters.recherche.enac.fr`.

In 2015, Lehouiller et al. [24] also proposed a general framework by modeling the problem with a graph where the vertices are the trajectories and the edges connect compatible trajectories. This is possible because the problem only involves binary constraints, i.e. constraints that specify the allowed combination of maneuvers for exactly two aircraft. The problem can thus be viewed as minimizing the cost of a maximum clique. For example in figure 1, each aircraft must choose between four maneuvers $\{a, b, c, d\}$. The edges represent compatible maneuvers. There is only one maximum clique representing a solution: $\langle 1d, 2c, 3c, 4b \rangle$. Lehouiller obtains good results using this model on problems involving up to 20 aircraft with a small number of maneuver options. Then in 2017, Lehouiller et al. [25] have proposed two decomposition algorithms to enhance the resolution. This graph model can easily be generated with our framework, but uncertainties are not taken into account. The same year, [31] proposed a new complex number formulation and convex relaxation for the centralized problem and showed that it could reduce the resolution time.

As observed from previous studies, classic algorithmic approaches of the en-route conflict resolution problem have not proved to be effective as soon as instances are of large dimension. In the last ten years, combining various algorithmic strategies, such as mathematical programming techniques and metaheuristics, has proved powerful in many domains [30]. Problems for which pure traditional approaches were uneffective could be successfully solved by exploiting synergies between different techniques [12] [11]. Even among metaheuristics, some are better at local search [26], while others cope well with global search [27]. Therefore, the hybridization of optimizers can alleviate the intrinsic drawbacks of basic algorithms [9], leading to reduced calculations, improvement of the precision of results and more stable convergence behaviors [15, 22]. The purpose of this paper is to show that combining different algorithms on the conflict resolution problem can improve the results obtained with pure approaches alone.

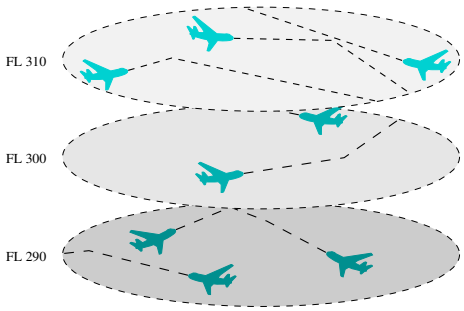


Figure 2: A traffic scenario on several Flight Levels.

3. Model

In this section, we present a generic approach to the conflict resolution problem which can be based on any conflict prediction system that takes as input a discrete set of possible maneuvers for each aircraft.

We first describe a possible model of realistic 3D maneuver options compatible with current Air Traffic Control (ATC) practice. Then we present a trajectory prediction model that approximates an aircraft possible positions at each time step as a convex polyhedron, according to a set of uncertainty parameters, as described in [4]. Possible conflicts are thereupon detected by an efficient GPU-based parallel algorithm described in [33], which improves the execution time by orders of magnitude compared to our previous sequential approach (cf. [4]). Eventually, the resulting *conflict matrix* is used to specify the constraints of the instance and a simple cost function is defined to favour later, shorter and least-deviating maneuvers.

3.1. Maneuvers and Decision Variables

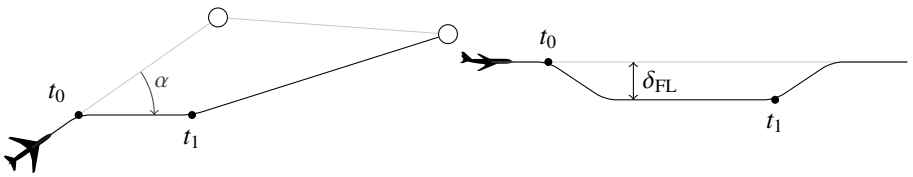
In the traffic scenarios used for our benchmark, aircraft are initially leveled on consecutive Flight Levels (FL) spaced-out by 1000 ft, as sketched in Figure 2. On each FL, the routes of the flight plans are defined by a sequence of waypoints (specified by their coordinates in the horizontal plane).

In our trajectory model, a maneuver could be a *heading* change (cf. Figure 3a), a *FL* change (cf. Figure 3b) or a speed change (cf. Figure 3c). These types of maneuvers are representative of ATC practice and can be easily implemented by pilots and current Flight Management System (FMS) technologies (cf. [19]). But we do not allow to combine different types in order to keep them simple enough.

The first phase of a maneuver begins at a discrete given time t_0, when it deviates from the initial trajectory, and its second phase starts at a later given time t_1, when the aircraft returns to its initial trajectory, as depicted in Figure 3. To recover the trajectory, described as a sequence of waypoints (depicted by large white dots in Figure 3a), while implementing an heading change maneuver, the first of the next waypoints that can be reached with an acceptable turning angle (i.e. $\leq 60^\circ$) is selected; therefore, some waypoints might be skipped as illustrated by Figure 3a. For vertical maneuvers, the

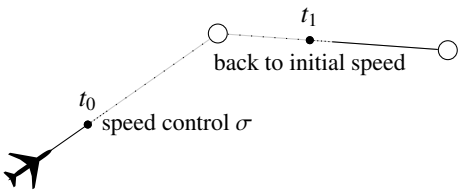
aircraft begin to climb or descend at a standard rate at time t_0 towards its assigned FL and starts to return to its original FL (also at a standard rate) at time t_1. Note that the time spent by an aircraft to climb or descend only depends on the climbing or descending rate and is not related to t_0 or t_1.

Heading changes α can take $n_\alpha = 6$ different values in our benchmark (see Section 5.1), i.e. 10, 20 or 30 degrees to the left or the right of the current heading. Vertical moves δ_{FL} can take $n_{\text{FL}} = 4$ values, i.e. climb or descend 1000 ft or 2000 ft (i.e. one or two FLs) from the current level. Speed changes σ can take $n_\sigma = 2$ different values, i.e. a $-6\,\%$ slowdown or a $+3\,\%$ acceleration (w.r.t. the current speed), which corresponds to the speed adjustment range deemed acceptable w.r.t. pilots and air traffic controllers constraints by the ERASMUS project [8]. The number of maneuver types is thus $n_k = n_\alpha + n_{\text{FL}} + n_\sigma = 12$. We limit the number of possible maneuvers by choosing t_0 among n_0 values (typically $n_0 = 4$ in our experiments). The number of values for t_1 is also chosen among a limited set of n_1 values (typically $n_1 = 4$).



(a) Horizontal maneuver.

(b) Vertical maneuver.



(c) Speed maneuver.

Figure 3: Maneuvers, beginning at t_0 and returning at t_1, compatible with current ATC practice. Large white dots correspond to waypoints of the initial trajectory, which is itself depicted in light gray. In figure 3c, the dotted line represents the parts of the trajectory where speed change is active, with the densely dotted sections corresponding to situations where the aircraft speeds up or slows down to the target speed.

If we combine n_0 values for t_0 and n_1 for t_1 with n_α possible angles, n_{FL} vertical maneuvers or n_σ possible speed modifications, plus one maneuver for unaltered aircraft (as a null heading, level change or speed change correspond to the same trajectory, regardless of t_0 and t_1), the number of maneuvers per aircraft is:

$$\begin{aligned} n_{\text{man}} &= n_0 \times n_1 \times (n_\alpha + n_{\text{FL}} + n_\sigma) + 1 \\ &= n_0 \times n_1 \times n_k + 1 \end{aligned}$$

Table 1 sums up the maneuver parameters and their respective values in the benchmark presented in Section 5.1, which amounts to $n_{\text{man}} = 4 \times 4 \times (6 + 4 + 2) + 1 = 193$

Table 1: Maneuver parameters.

Parameter		Size (value)		Typical values
t_0	start time	n_0	(= 4)	0, 1, 2 and 3 min
t_1	return time	n_1	(= 4)	5, 6, 7 and 8 min
α	heading change	n_α	(= 6)	-30, -20, -10, 10, 20 and 30°
δ_{FL}	FL change	n_{FL}	(= 4)	-20, -10, 10 and 20 FL
σ	speed change	n_σ	(= 2)	-6 and $+3$ %

combinations. For a conflict cluster with n aircraft, the search space size is then:

$$n_{\text{man}}^n$$

therefore $193^{20} \approx 5.14 \times 10^{45}$ for a 20-aircraft instance and up to 1.36×10^{137} for 60 aircraft.

To provide a generic view of the maneuver model, we restrict the number of parameters for each aircraft i to a single decision variable m_i that aggregates variables t_0, t_1 and the heading change α, the FL change δ_{FL} or the speed change σ, thanks to a bijection from the valid 5-tuples to interval $\{1, \ldots, n_{\text{man}}\}$. We call $\mathcal{M}$ the set of decision variables of the problem:

$$\mathcal{M} = \{m_i \in \{1, \ldots, n_{\text{man}}\}, \ \forall i \in \{1, \ldots, n\}\} \tag{1}$$

3.2. Trajectory Prediction and Conflict Detection

To compute possible conflicts between maneuvers within the time frame of the resolution, we first model various sources of uncertainty, then describe how our trajectory prediction take them into account to detect incompatible maneuvers and build the *conflict matrix*.

3.2.1. Conflict

For ATC, a conflict occurs between two aircraft if there is a simultaneous loss of horizontal and vertical separation according to some distance thresholds (called *separation norms* in the following) which depend on the airspace considered:

Definition 1 (Conflict Between Aircraft). *Aircraft i and j are in* conflict *iff $\exists t$ s.t.:*

$$dist_h(p_i(t), p_j(t)) \leq \text{norm}_h \quad \wedge \quad dist_v(p_i(t), p_j(t)) \leq \text{norm}_v$$

where:

- $p_k(t) = (x_k^t, y_k^t, z_k^t)$ *is the position of aircraft k at time t;*
- $dist_h(p_i(t), p_j(t)) = \sqrt{(x_i^t - x_j^t)^2 + (y_i^t - y_j^t)^2}$ *is the distance in the horizontal plane;*
- $dist_v(p_i(t), p_j(t)) = |z_i^t - z_j^t|$ *is the distance between altitudes.*

For en-route traffic, the separation norms are usually $\text{norm}_h = 5\,\text{NM}$ and $\text{norm}_v = 1000\,\text{ft}$ as illustrated by Figure 4 showing the *protection volume* around an aircraft. A conflict occurs whenever another intruding aircraft enters the protection volume.

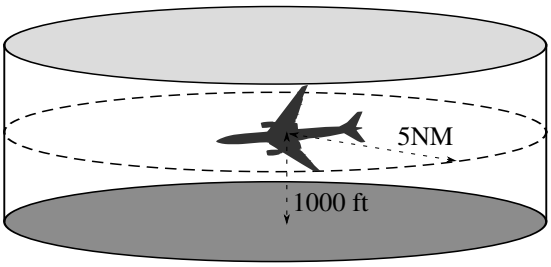


Figure 4: Typical en-route protection volume around an aircraft.

3.2.2. Uncertainties on Trajectories

To model the inaccuracy of realistic trajectory prediction systems, we model the six following sources of uncertainties, associated to the implementation of the maneuver or to the state of the aircraft. As in Figure 3, large white dots represent waypoints of the aircraft flight plan, while small black dots correspond to turning points of maneuvers in Figures 5–9.

- When instructed to maneuver, a pilot can react more or less quickly. Uncertainty $\varepsilon_{t_0} \in [0, E_{t_0}]$ representing the maximum reaction time to start a maneuver is associated to time t_0 (see Figure 5).
- Uncertainty $\varepsilon_{t_1} \in [0, E_{t_1}]$ representing the maximum reaction time for ending a maneuver is associated to time t_1 (see Figure 5).
- Uncertainty $\varepsilon_\alpha \in [-E_\alpha, E_\alpha]$ is also associated to the heading change angle α (see Figure 6).
- Horizontal aircraft speeds v_h are subject to relative error $\varepsilon_{v_h} \in [-E_{v_h}, E_{v_h}]$ (expressed as a percentage) such that future positions of aircraft are spread over a range which grows with time (see Figure 7).
- Climbing and descending rates v_v are also subject to relative error $\varepsilon_{v_v} \in [-E_{v_v}, E_{v_v}]$ (as a percentage) as illustrated in Figure 8.
- The *fly mode* f_m can be chosen among two values $f_m \in \{F_b, F_o\}$ as an aircraft can "fly by" (F_b) or "fly over" (F_o) a waypoint, depending on the pilot practice or the airline rules, and we consider both options to build the future trajectory (see Figure 9). More precisely, when an aircraft must turn at a waypoint, it cannot strictly fly linear segments with instant turning points. Flight Management Systems or pilots can either "fly by" or "fly over" the turning point: when the pilot anticipates the turning angle before arriving at the waypoint, she flies by the waypoint, and when the pilot turns once she has reached the waypoint and heads back to the initial trajectory after it, she flies over the waypoint. Because we do not know which choice is going to be made by the pilot, we take the so-called "fly mode" uncertainty into account in our model.

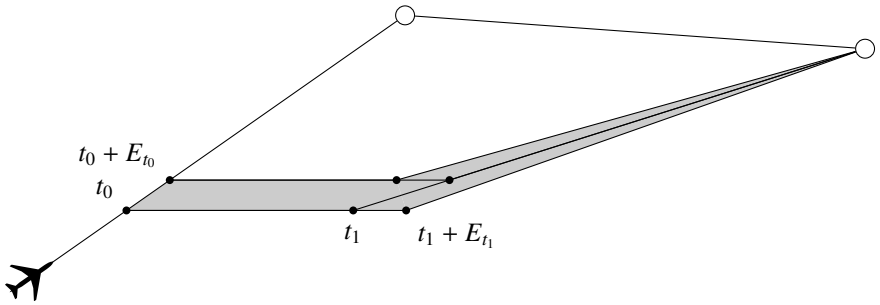


Figure 5: Reaction time uncertainty model with maximal errors E_{t_0} and E_{t_1}.

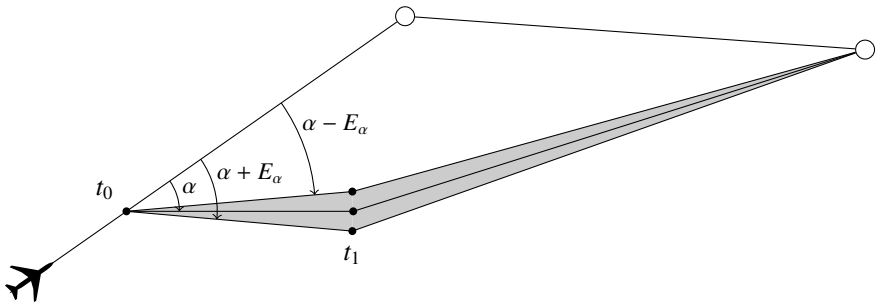


Figure 6: Heading change uncertainty model with maximal error E_α.

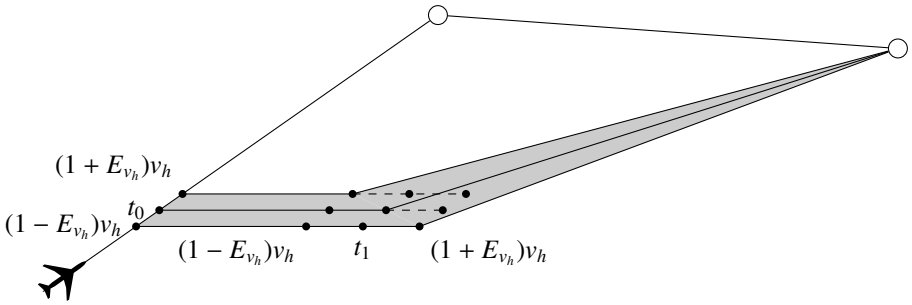


Figure 7: Speed uncertainty model with maximal error E_{v_h}.

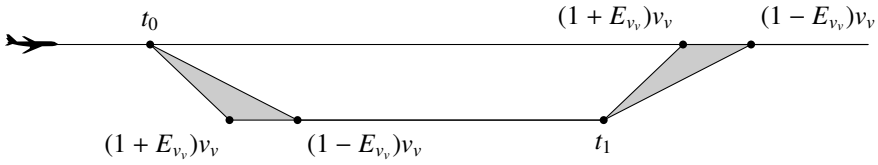


Figure 8: Climb and descent uncertainty model with maximal error E_{v_v} model.

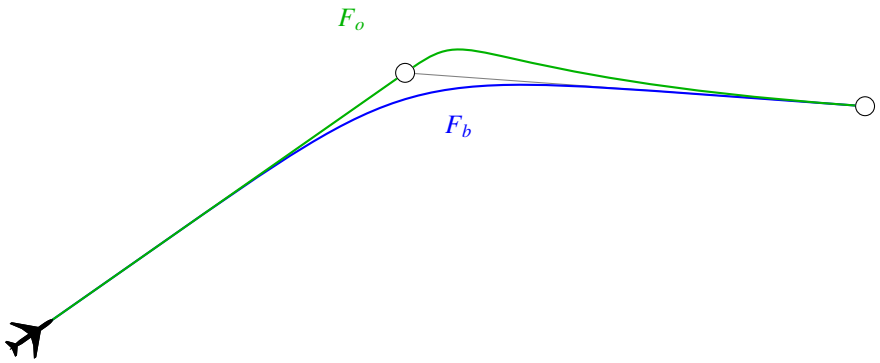


Figure 9: Flight mode uncertainty model with possible modes "fly by" (F_b) or "fly over" (F_o).

Table 2: Uncertainties on the trajectory parameters.

Parameter		Error	Typical values
t_0	start time	$\varepsilon_{t_0} \in [0, E_{t_0}]$	10, 20 and 30 s
t_1	return time	$\varepsilon_{t_1} \in [0, E_{t_1}]$	10, 20 and 30 s
α	angle	$\varepsilon_\alpha \in [-E_\alpha, E_\alpha]$	1, 2 and 3°
v_h	horizontal speed	$\varepsilon_{v_h} \in [-E_{v_h}, E_{v_h}]$	2, 4 and 6 %
v_v	vertical speed	$\varepsilon_{v_v} \in [-E_{v_v}, E_{v_v}]$	5, 10 and 15 %
f_m	waypoint fly mode	$f_m \in \{F_b, F_o\}$	F_b, F_o

Table 2 sums up these uncertainties, related to the maneuver parameters at the top of the table, and to the aircraft characteristics only at the bottom. Note that there is no uncertainty on the lateral position when an aircraft is heading toward a waypoint as its FMS is able to dynamically correct the lateral error. Accordingly, no uncertainty is considered for the maneuver parameter specifying the FL change δ_{FL} as current FMS are able to precisely level out at the specified FL. Uncertainty on the vertical profile is therefore taken into account by the error on the vertical speed ε_{v_v} alone.

3.2.3. Aircraft Position Envelope

To detect the conflicts, time can be discretized into regular steps, provided their duration τ is small enough to avoid missing even the shortest conflicts. In Section 5.1, we fix $\tau = 3$ s because two facing aircraft flying at 600 kn (maximal speed for airliners) get only 1 NM closer every 3 s, which ensures the detection of such a conflict, as the target separation distance is 5 NM (see [10] for a more in-depth discussion on this topic).

We assume that at a given time step, the position of an aircraft belongs to the set of positions allowed by any combination of the uncertainty parameters with a uniform probability distribution (but this assumption could be refined with a more realistic distribution model as described in [16]). We therefore build the envelopes of possible positions at each time step for all maneuvers before checking their distance to detect a possible conflict (see Section 3.2.4).

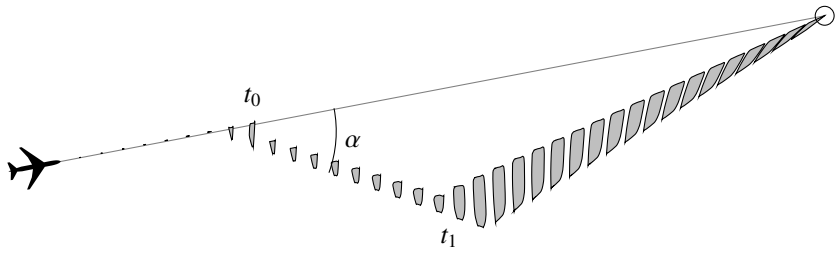


Figure 10: An example of convex hulls representing a maneuver with uncertainties in the horizontal plane at each time step.

As described in Section 3.1, we have defined six uncertainty parameters for our trajectory prediction: ε_{t_0}, ε_{t_1}, ε_{α}, ε_{v_h}, ε_{v_v} and f_m. In order to take into account every possible trajectory, we test every combination of the extreme values of these parameters: $2^6 = 64$ trajectories are computed to built the geometric boundaries of a single maneuver over time. Once these extreme trajectories are built, we compute for each time step t a 3D polyhedral convex envelope to safely approximate the possible positions of an aircraft. We use the well-known Graham's algorithm [18] to build the corresponding smallest convex hull in the horizontal plane, and take the minimum and maximum altitudes in the vertical plane. The resulting prism is the smallest convex orthogonal cylinder that includes the set of the possible positions of the aircraft at step t.

Figure 10 gives an example in the horizontal plane of a 30° heading change maneuver starting at $t_0 = 5\,\text{min}$, lasting 10 min (therefore $t_1 = 15\,\text{min}$), with $E_{t_0} = E_{t_1} = 60\,\text{s}$, $E_{\alpha} = 5°$ and $E_{v_h} = 5\,\%$. The original route of the aircraft is represented as a gray solid line and the maneuver is depicted every minute as a polygon corresponding to the convex hull that includes the possible aircraft positions.

3.2.4. Conflict Detection

In the previous section, each predicted trajectory is modeled as a sequence of aircraft position envelopes. As the notion of conflict is only defined pointwise between two perfect aircraft trajectories, we extend the notion of conflict to trajectories modeled as a sequence of aircraft position envelopes.

Definition 2 (Conflict Between Predicted Trajectories). *Predicted trajectories p and q are in* conflict *iff* $\exists$ *t,* $a \in p(t)$ *and* $b \in q(t)$ *s.t.:*

$$\text{dist}_h(a, b) \leq \text{norm}_h \quad \wedge \quad \text{dist}_v(a, b) \leq \text{norm}_v$$

where $p(t)$ *and* $q(t)$ *are aircraft position envelopes at time t, and a and b are 3D positions.*

Because of the uncertainties defined in section 3.2.2, when predicting aircraft trajectories, we define the future possible positions of each aircraft at a given time with a convex envelope that includes all the possible locations of the aircraft. In this context, Definition 2 generalizes Definition 1 by requiring that all possible positions of the first aircraft combined with all possible positions of the second aircraft satisfy

Definition 1. Therefore, two aircraft will be in conflict according to Definition 2 iff the two envelopes representing them at a given time are closer than 5 NM horizontally and 1000 ft vertically, even if some pairs of points within the two envelopes are not in conflict according to Definition 1. Note that the top and bottom faces of the 3D envelope corresponding to the exclusion volume of an aircraft do not become slanted during climb or descent, but remains horizontal, as only the minimal and maximal altitude are taken into account to approximate the possible positions of an aircraft as mentioned in the previous section.

We assume that the actual future trajectory will always be inside the aircraft position envelopes of the predicted trajectory. Thus, with Definition 2, if the predicted trajectories are not in conflict then this property will hold for the actual future trajectories as well.

Using this definition, we can compute all the conflicts between all the predicted trajectories considered for the resolution and store them in a 4D boolean matrix. As described in 3.1, maneuvers are numbered from 1 to n_{man}. Then, for each pair of aircraft (i, j), with $i < j$ as the conflict relation is symmetric, and each pair of maneuver options (k, l), where k is a maneuver option for aircraft i and l for aircraft j, we test if maneuvers k and l are in conflict to set the coefficients $C_{i,j,k,l}$ of the conflict matrix:

Definition 3 (Conflict Matrix). *The coefficients of the symmetric conflict matrix of dimensions $n \times n \times n_{\text{man}} \times n_{\text{man}}$ are defined by:*

$$C_{i,j,k,l} = \begin{cases} 1 & \textit{if maneuver k and l conflicts according to Definition 2} \\ 0 & \textit{otherwise} \end{cases}$$

$\forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, n\}$ *s.t.* $i < j, \forall k \in \{1, \ldots, n_{\text{man}}\}, \forall l \in \{1, \ldots, n_{\text{man}}\}$.

The calculation of such a conflict matrix requires to compare $n_{\text{man}}^2 \frac{n(n-1)}{2}$ pairs of predicted trajectories. In the end, it leads to compare $n_{\text{man}}^2 \frac{n(n-1)}{2} T$ pairs of aircraft position envelopes where T is the number of time steps and therefore of aircraft position envelopes per trajectory. This number can be huge even with a limited number of aircraft. For the smallest scenario in this paper with $n = 15$ aircraft, $n_{\text{man}} = 193$ possible trajectories and $T = 150$ time steps, we have to check the separation for 586 672 750 envelope pairs. If we were to integrate our conflict solver in a traffic simulator or operational system, we would have to solve iteratively a sequence of conflict resolution problems over a Rolling Horizon (RH) with the fastest possible update rate allowed by the running time of the resolution process (see [19]). Therefore, the conflict matrices must also be built as fast as possible.

In order to avoid numerous time-consuming distance computations, we can use simple bounding volumes with cheap intersection checks. As depicted by Figure 11, for each aircraft position envelope we can compute an Axis-Aligned Bounding Box (AABB) increased by half the separation norm (cf. Definition 1). The intersection test between AABBs is very fast as it requires six floating-point number comparisons at most. If the AABBs of two aircraft position envelopes do not intersect then the envelopes cannot conflict. Otherwise, we use the ISA-GJK algorithm [34], a more time-consuming test, to check if the envelopes intersect. It is a variant of the GJK algorithm [17] which is widely used in robotics and computational geometry to determine the distance between two convex shapes. This algorithm runs in linear time w.r.t. the number of vertices of the aircraft position envelopes.

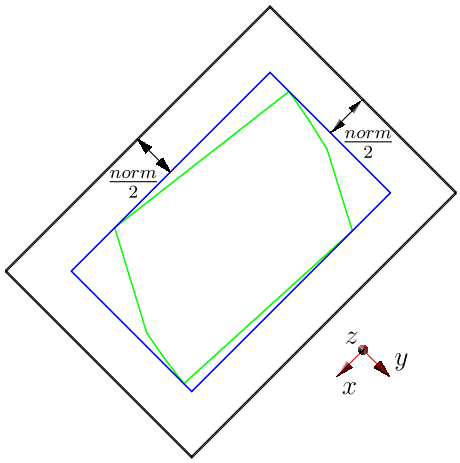


Figure 11: Axis-Aligned Bounding Box (in black) of an aircraft position envelope (in green).

To reduce even more the running time of our solver, the computation of the conflicts can be parallelized. A parallel implementation on Graphics Processing Unit (GPU) described in [33] using the AABBs and ISA-GJK has been successfully tested to compute conflict matrices with a decrease of two orders of magnitude for the total running time of the detection phase. For instance, the conflict matrix of the largest scenario available in our online benchmark (see Section 2) with $n = 100$, $n_{\text{man}} = 193$ and $T = 150$ is computed in 1 s only. On average, the trajectory pairs are processed at a rate of 117 000 pairs per millisecond.

3.3. Cost of Maneuvers

To discriminate among the solutions to feasible instances, we build an arbitrary cost function for the maneuvers so as to ensure the following properties which characterize good solutions from an operational point of view:

1. any maneuver is more costly than no maneuver;
2. maneuvers should start as late as possible: because uncertainties are reduced over time for successive resolution problems in a RH solver (cf. Section 3.2.4), the cost of a delayed maneuver could be reduced when the problem is updated;
3. maneuvers should be as short as possible;
4. the angle should be as small as possible;
5. FL change should be as small as possible;
6. a $-6\,\%$ slowdown or $+3\,\%$ acceleration is deemed equivalent to a 10° heading change (for the sake of simplicity);
7. a 1000 ft vertical maneuver is considered equivalent to a 20° heading change, and a 2000 ft one to 30°.

To compute the cost of a maneuver, values of t_0 are enumerated by an index $k_0 \in \{1, ..., n_0\}$, values of t_1 by index $k_1 \in \{1, ..., n_1\}$ and angles α, of value 10, 20 or 30

degrees right or left, are respectively indexed by $k_\alpha \in \{1, ..., \frac{n_\alpha}{2}\}$. Speed changes σ are emunerated by an index $k_\sigma \in \{1, ..., \frac{n_\sigma}{2}\}$. Similarly, FLs are indexed by $k_{\text{FL}} \in \{1, ..., \frac{n_{\text{FL}}}{2}\}$. For our benchmark, the cost of a maneuver is then defined as follows:

Definition 4 (Cost of a Single Maneuver). *The cost $c(m_i)$ of a single maneuver $m_i \in \{1, ..., n_{\text{man}}\}$ for aircraft i is:*

$$c(m_i) = \begin{cases} (n_0 - k_0)^2 + k_1^2 + k_\alpha^2 & \text{if } \alpha \neq 0 \\ (n_0 - k_0)^2 + k_1^2 + (1 + k_{\text{FL}})^2 & \text{if } \delta_{\text{FL}} \neq 0 \\ (n_0 - k_0)^2 + k_1^2 + k_\sigma^2 & \text{if } \sigma \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

where k_0, k_1, k_α, k_{FL} and k_σ are the indices corresponding to the values of t_0, t_1, α, δ_{FL} and σ for maneuver m_i.

Note that the cost is null whenever an aircraft is not maneuvered.

As this work is based on a generic framework that separates the solver method from the problem itself, we chose to instantiate our benchmark with the most simple and intuitive cost function we could come up with, easy to understand and reproduce, so as to encourage the comparison of different solvers by interested readers, which might be unfamiliar with all the quirks of operational ATC costs. In a real environment, it should be modified to comply with aircraft performance models on one side and controllers' preferences on the other side. We could easily replace this cost by a more accurate one, taking into account the fuel consumption, or pilots and air traffic controllers preferences. This would however not change the model but only the values given to each maneuver of each aircraft. The same resolution methods would apply but provide different results.

Given an instance with n aircraft, we define the cost of a solution as the sum of the costs of the maneuvers for all aircraft:

Definition 5 (Cost of Conflict Resolution). *The cost of a solution to a conflict resolution problem with n aircraft is:*

$$\text{cost}(\mathcal{M}) = \sum_{i=1}^{n} c(m_i)$$

Note that $\mathcal{M}$ is the set of decision variables of the problem defined in 1.

3.4. Overall Mathematical Model

Eventually, we model conflict resolution as the following combinatorial optimization problem which summarizes the preceding sections.

Decision variables. For a problem with n aircraft, the set of decision variables is:

$$\mathcal{M} = \{m_i \in \{1, \ldots, n_{\text{man}}\},\ \forall i \in \{1, \ldots, n\}\}$$

where the maneuver of aircraft i is represented by decision variable m_i and all the maneuver options associated with allowed tuples $\langle t_0, t_1, \alpha, \delta_{\text{FL}}, \sigma \rangle$ are numbered from 1 to n_{man} as described in Section 3.1. Hence, the size of the search space is $n^{n_{\text{man}}}$. Note that different sets of possible maneuvers could also be specified for each aircraft without loss of generality.

Constraints. According to the 4D conflict matrix C defined in Section 3.2, for each element $C_{i,j,k,l} = 1$, maneuvers k of aircraft i and l of aircraft j cannot be chosen at the same time. The constraints of our problem are therefore defined by:

$$m_i \neq k \vee m_j \neq l,$$
$$\forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, n\} \text{ s.t. } i < j, \forall k \in \{1, \ldots, n_{\text{man}}\}, \forall l \in \{1, \ldots, n_{\text{man}}\} \text{ s.t. } C_{i,j,k,l} = 1$$

Cost. The cost of an optimal solution to a conflict resolution problem is equal to:

$$\min_{\forall i \in \{1,\ldots,n\}, m_i \in \{1,\ldots,n_{\text{man}}\}} \sum_{i=1}^{n} c(m_i)$$

with

$$c(m_i) = \begin{cases} (n_0 - k_0)^2 + k_1^2 + k_\alpha^2 & \text{if } \alpha \neq 0 \\ (n_0 - k_0)^2 + k_1^2 + (1 + k_{\text{FL}})^2 & \text{if } \delta_{\text{FL}} \neq 0 \\ (n_0 - k_0)^2 + k_1^2 + k_\sigma^2 & \text{if } \sigma \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

where maneuver m_i corresponds to allowed tuple $\langle t_0, t_1, \alpha, \delta_{\text{FL}}, \sigma \rangle$ and $k_\star \in \{1, \ldots, n_\star\}$ indexes the possible values associated with $\star \in \{0, 1, \alpha, \text{FL}, \sigma\}$ (a generic parameter subscript) by increasing absolute value (as described in Section 3.3), such that later, shorter, and least-deviating maneuvers are favoured. Note that any cost function, possibly customized for each aircraft, could also be used in our model.

4. Resolution Algorithms

In this section, we first present two methods to solve the conflict resolution problem, then describe how they can cooperate to improve the performances of our solver.

The first one, a *Memetic Algorithm* (MA), is a metaheuristic that mimics natural evolution to explore the search space and can find optimal solutions for reasonable instances in a short amount of time. However, for larger instances, a MA may remain stuck in local optima and fail to discover an optimal solution. Moreover, metaheuristics are not exact algorithms, hence they can neither prove the optimality of a solution nor the infeasibility of an instance.

The second one, *Integer Linear Programming* (ILP), is based on the *Branch and Cut* algorithm and able to quickly prove optimal solutions (or infeasibility) for small instances. Though for large instances (more than 50 aircraft), the resolution is too time-consuming for a real-time system.

To benefit from the advantages of both combinatorial solvers and find better solutions to large instances, our third approach consists in their *cooperation*. We first present a distributed framework that can leverage any combinatorial solver with the suitable interface. Then we describe how to adapt a solver to take into account and provide information on the state of the search from and to other solvers.

4.1. Memetic Algorithm

In this section, we first present the general principle of our Memetic Algorithm (MA) and the objective function used to solve the conflict resolution problem. Then, we detail the Tabu Search included in our MA and the crossover operator used to recombine current solutions into new ones.

4.1.1. Principle

An MA, described in Algorithm 1, is a hybridization of an Evolutionary Algorithm and a Tabu Search (TS) such as presented in [21]. The main feature of an MA is that each element of the population is a local minimum.

Algorithm 1 Memetic algorithm (MA)

1: population ← `initializePopulation`()
2: **while** termination criterion is not met **do**
3: (parent$_1$, parent$_2$) ← `select`(population)
4: child ← `crossover`(parent$_1$, parent$_2$)
5: (child, c_c) ← `tabuSearch`(child)
6: population ← `replace`(population, child, c_c)
7: **end while**
8: Return the best element of population

First (line 1), a population of candidate solutions is randomly initialized and a TS (described in Algorithm 2) is applied to each candidate. Then (line 3), we randomly select two elements in the population called *parents* and generate a new element called a *child* through a standard *crossover* operation between the two parents that recombines their maneuvers (line 4). Afterwards (line 5), the child is improved by applying a TS until a local minimum is found (with cost c_c). Then it replaces the worst element of the population if its cost is lower and if it does not already belong to the population (line 6). We iterate this procedure until a given time limit (line 2) or when no improvement is made for a given number of iterations.

4.1.2. Objective Function

The objective function of our MA represents the function to minimize. Here, we first focus on finding a conflict-free set of maneuvers, with the smallest cost as a secondary objective. Therefore, we define the objective function as the linear combination of two terms, the number of remaining conflicts and the cost of a solution (cf. Definition 5):

$$f(M) = P \times \sum_{\forall m_i, m_j \in M \text{ s.t. } i<j} C_{i,j,m_i,m_j} + \text{cost}(M)$$

where P is a big (enough) integer to guarantee that the cost of a solution is always higher than the one of another solution if it has more conflicts.

4.1.3. Improvement of New Candidates with Tabu Search

Each generated solution candidate (either at initialization or after a crossover operation) is improved with respect to the objective function by a TS, as described in Algorithm 2. A TS, like all local search algorithms, modifies the current solution by successive small changes called *moves*. The moves used in our TS consist in the modification of the maneuver assigned to one of the aircraft, which defines the *neighborhood* of a candidate solution called "1-move" neighborhood.

First (line 4), we select the best move *mv* from the neighborhood of the current solution that is not in tabuList, i.e. the list of forbidden maneuvers for each aircraft

Algorithm 2 Tabu Search (TS)

1: **Input :** s, a candidate solution
2: tabuList $\leftarrow \varnothing$
3: **while** termination criterion is not met **do**
4: $mv \leftarrow$ `selectBestMove`$(s, \text{tabuList})$
5: $s \leftarrow$ `move`(s, mv)
6: tabuList $\leftarrow$ `update`$(\text{tabuList}, $`reverse`$(mv))$
7: $(s_{\text{best}}, c_{\text{best}}) \leftarrow$ `saveBest`(s_{best}, s)
8: **end while**
9: Return $(s_{\text{best}}, c_{\text{best}})$

during a given number of iterations (which is initially empty, cf. line 2). Notice that in a TS, the selected move is not necessarily a move that improves the cost of the candidate solution, because our TS has two different successive phases involving different roles during resolution as to mentioned further. In a second step (line 5), we update the current solution by applying the selected move and add its reverse to tabuList (line 6) to prevent the change to be undone during a given amount of iterations. Finally (line 7), we update the best solution and its cost if needed. We iterate this procedure until a fixed number of iterations (line 3).

Given the objective function, $f(M)$, there are two different successive phases in our TS process:

Phase 1 When the candidate solution s still involves some remaining conflicts, i.e. s is not an admissible solution (therefore $\sum_{\forall m_i, m_j \in M \text{ s.t. } i<j} C_{i,j,m_i,m_j} > 0$), our TS only minimizes the number of remaining conflicts. The purple track in Figure 12 represents this first phase of our TS inside the solutions landscape.

Phase 2 When the candidate solution s is a conflict-free set of maneuvers, i.e. s is an admissible solution (therefore $\sum_{\forall m_i, m_j \in M \text{ s.t. } i<j} C_{i,j,m_i,m_j} = 0$), then our TS minimizes the cost of the solution cost(M) without creating new conflicts. The blue track in Figure 12 represents this second phase of our TS. Notice that during this second phase, the candidate solution s cannot step outside its admissible configuration area (represented by green circles in Figure 12). Indeed, the admissible configuration set is not a connected space if we consider the 1-moves neighborhood.

As a result, if only the TS is used, each of its run will find a solution trapped in an admissible configuration area. Moreover, the number of admissible configuration areas might be exponential in the problem size. In such a case, even a multi-start strategy on top of the TS would be inefficient, so we decided to use a crossover operator to be able to escape the admissible configuration areas while maintaining relatively structured solution candidates.

4.1.4. Crossover

We have used a standard crossover operator, named *uniform crossover*, that generates a single new candidate solution from two elements of the population. For each aircraft,

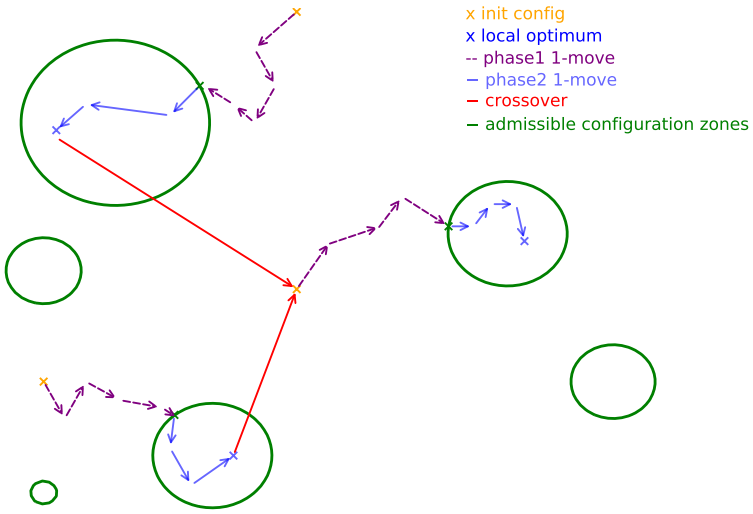


Figure 12: Diagram representing the behavior of our MA in the landscape of solutions. The inside of a green circle corresponds to a subset of the admissible solutions.

this crossover operator randomly selects one of the two maneuvers of the parents and assigns it to their child.

The objective function value of a child is generally greater than those of its parents as the crossover may generate many new conflicts. However, a crossover is systematically followed by a TS to "repair" the child and mitigate its poor quality. Actually, the crossover operator essentially has a role of diversification in an MA, i.e. exploring uncharted part of the search space to avoid being stuck in local minima. In Figure 12, the crossover of two admissible solutions that enables to explore a new admissible configuration area is represented in red. Other crossover operators, such as *one-point crossover* or *k-point crossover*, have been tested on our benchmark but with no better results.

4.2. Integer Linear Programming

Integer Linear Programming (ILP) [23] has become a very powerful tool for modeling and solving real-world combinatorial optimization problems, like planning and scheduling. The main resolution algorithm, *Branch and Cut* (BC) [28], uses a *simplex* algorithm to solve continuous relaxations of the problem and add cutting planes at each node of a *Branch and Bound* search. State of the art ILP solvers like Gurobi [20], which we used to obtain the results presented in Section 5, significantly improve the efficiency of the basic BC algorithm with preliminary transformation techniques to reduce the size of combinatorial problems, as well as heuristics to obtain better objective bounds

during the search. These sophisticated refinements fall beyond the scope of this paper and inquisitive readers may refer to [1] to obtain more information.

In the following section, we first present a basic ILP model for conflict resolution, then we show how to obtain an equivalent but much more compact and efficient "aggregated" model.

4.2.1. Basic Model

In this section, we describe a straightforward ILP model of the en-route conflict resolution problem with binary assignment variables. First, the assignment binary decision variables are defined by:

$$\forall i \in \{1,\ldots,n\}, \forall k \in \{1,\ldots,n_{\text{man}}\}, \quad x_{i,k} = \begin{cases} 1 & \text{if flight } i \text{ is assigned to maneuver } k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

As each flight must choose exactly one maneuver, we had the corresponding constraint for each flight:

$$\sum_{k=1}^{n_{\text{man}}} x_{i,k} = 1, \quad \forall i \in \{1,\ldots,n\} \tag{3}$$

Moreover, conflicting trajectories cannot be chosen simultaneously:

$$x_{i,k} + x_{j,l} \leq 1, \quad \forall i,j \in \{1,\ldots,n\} \text{ s.t. } i < j, \forall k,l \in \{1,\ldots,n_{\text{man}}\} \text{ s.t. } C_{i,j,k,m} = 1 \tag{4}$$

As the cost of a maneuver does not depend on the aircraft in our model, the objective function to minimize can be expressed as the sum of the products of the cost $c(k)$ of each maneuver k by the sum of the corresponding binary decision variables $x_{i,k}$:

$$\min \sum_{k=1}^{n_{\text{man}}} c(k) \times \sum_{i=1}^{n} x_{i,k} \tag{5}$$

Note that specific maneuver costs $c(i,k)$ for each aircraft could easily be taken into account by generalizing equation 5 with: $\sum_{i=1}^{n} \sum_{k=1}^{n_{\text{man}}} c(i,k) x_{i,k}$.

In this basic model, the conflict constraints 4 are straightforward but the resulting number of equations is in $O(n^2 \times n_{\text{man}}^2)$, which can be huge for large numbers of aircraft and maneuvers. For example, with an instance with 50 aircraft and 193 maneuvers, there will be more than 9.3×10^7 constraints.

4.2.2. Aggregated Model

To avoid the large number of constraints (quadratic with n and n_{man}) of the previous basic model, it is possible to aggregate, for a given trajectory k of a given aircraft i, all conflicting trajectories belonging to aircraft with indices $j > i$ into a single constraint:

$$\sum_{j=i+1}^{n} \sum_{l=1}^{n_{\text{man}}} C_{i,j,k,l} x_{j,l} \leq n_{i,k}(1 - x_{i,k}), \quad \forall i \in \{1,\ldots,n\}, \forall k \in \{1,\ldots,n_{\text{man}}\} \tag{6}$$

with $n_{i,k} = |\{j \in \{i+1,\ldots,n\} \text{ s.t. } \sum_{l=1}^{n_{\text{man}}} C_{i,j,k,l} \geqslant 1\}|$, the number of aircraft (with indices greater than i) that have at least one maneuver in conflict with maneuver k of aircraft i.

If trajectory k of aircraft i is chosen, i.e. $x_{i,k} = 1$, the right-hand side of constraint 6 becomes 0, so all the $x_{j,l}$ conflicting with it (i.e. $C_{i,j,k,l} = 1$) on the left-hand side must also be assigned 0, as all the corresponding trajectories conflict with trajectory k of aircraft i. Otherwise, $x_{i,k} = 0$ and the constraint is relaxed such that any aircraft j can choose a trajectory conflicting with trajectory i of aircraft k. Conversely, if any conflicting $x_{j,l}$ of the left-hand side is assigned 1, then $x_{i,k}$ must be assigned 0 as trajectory k of aircraft i can no longer be chosen.

Combined with constraints 3 and objective 5, this aggregated model is equivalent to the basic model described in the previous section, but the number of constraints is reduced by orders of magnitude as there are only $O(n \times n_{\text{man}})$ constraints instead of $O(n^2 \times n_{\text{man}}^2)$. If we still take 50 aircraft and 193 maneuvers into account, there will be only 9650 constraints (omitting the 50 constraints 3). As this model consistently outperformed the basic one by orders of magnitude in our experiments, the results presented in section 5 were all obtained using the aggregated model.

4.3. Cooperation Between Memetic Algorithm and ILP Solver

We presented the MA metaheuristic in Section 4.1 and an exact ILP algorithm in Section 4.2, two classic techniques to solve Combinatorial Optimization Problems (COP). If exact algorithms are able to find and prove optimal solution for every instance of a COP (or its infeasibility), their running times, however, increase dramatically with the problem size as shown in Section 5. For large instances, especially at the beginning of the resolution, metaheuristics trade optimality for execution time as they are designed to obtain good but not necessary optimal solutions in very limited time and without optimality proof. However, metaheuristics generally visit the same solution more than once, which costs computation time.

When measuring the advantages and drawbacks of exact and metaheuristic algorithms, especially in terms of performances on the conflict resolution problem, both approaches can be seen as complementary. On the one hand, as mentioned in [30], in order to keep the BC search tree relatively small, high-quality solutions and bounds are very useful. So BC can benefit from the ones provided by a metaheuristic, especially at the beginning of the resolution. On the other hand, metaheuristics can generally get stuck in local optima, which might prevent them from converging to the global optimum. Therefore, they can also benefit from improved solutions shared by the BC algorithm to explore better parts of the search space.

Several ways exist to hybridize mathematical programming techniques (e.g. ILP) and metaheuristics [11]. The most traditional approach is to use metaheuristics for providing high-quality solutions inside a BC algorithm. On the other hand, a frequently applied approach is Large Neighborhood Search exploration thanks to ILP algorithm. Here, we use a cooperation approach to run the MA and ILP solvers simultaneously and both algorithms can share information during the resolution process with the other one.

In the following sections, we first specify a distributed system which can integrate any combinatorial solver with the suitable interface in Section 4.3.1, and Section 4.3.2 describes the adaptation of the MA and the BC algorithm to fit our cooperation approach.

4.3.1. Distributed System for Cooperation

Our distributed system is composed of a central process that acts as a data manager and any number of solvers that share data through this central process. Currently, the managed data consist of full solutions, lower bounds for the minimization criterion and the status of the optimality proof. However, the framework is completely generic, thus any useful information could be exchanged (e.g. new global cuts found by one of the solvers). At any time, each solver has the ability to post (possibly) new information, or to retrieve the current best information for its internal use.

Given this general pattern, we chose to implement it as a client-server architecture, where the data manager acts as the server and each solver is a client which connects to the server. The solutions and bounds are exchanged as messages between the server and clients. Besides its simplicity, this client-server architecture has the advantage of being distributed over any number of processors on a given network. Also, it is platform- and language-agnostic, as the only requirement is to be able to connect to the server with standard protocols.

The problem to solve might be modeled differently in the various clients (i.e. solvers): e.g. the n decision variables in our MA model directly represent maneuvers, whereas the $n \times n_{\text{man}}$ ILP ones only correspond to possible assignments. However, they must comply to the common interface proposed by the server to exchange information such as the upper and lower bounds or feasible solutions.

For example, once a client has found a new better solution, it will first translate the solution to the common interface then send the corresponding message to the server (i.e. post the solution to the data management process). When the server receives the message, it checks whether the information is useful, e.g. a feasible solution with a better cost than the best solution so far, or a higher lower bound than the current best one. If required, the server updates its own bound and current best solution. Upon request, each client will receive the latest bound and solution. Any solver that proves the optimality of a solution can also send this information to the server in order to end the search.

4.3.2. Collaborative Version of MA and BC

The original MA was modified to build the Collaborative Memetic Algorithm (CMA), see algorithm 3, which is able to exchange optimization information with other solvers. First (line 1), the CMA registers to the server and receives the problem to solve, as well as the current best solution s_r if any. If a feasible solution is received, it is then included in the population (line 4). After the tabu search (line 13), if the CMA finds a better solution, we update this solution and send it to the server. Then (line 15), CMA tries to obtain a better solution from the server (or even an optimality proof). If the optimality is not proved (line 16), but a better solution is received (line 17), then we include this new solution to improve the population (line 18). Otherwise, if the optimality has been proved by other clients, the search can be terminated.

Collaborative Branch and Cut (CBC), see algorithm 4, differs from the original version in several ways. First (line 1), as in the CMA, the CBC algorithm registers to the server and gets the problem and the current state of the search. If any solution s_r or lower bound lb have been received (line 4), they are updated. During the resolution process, any better feasible solution, once found by CBC, is sent to the server (line 15). Then

Algorithm 3 Collaborative Memetic algorithm (CMA)

1: $(P_0, s_r, \text{proof}) \leftarrow$ `register`(*server*) // get problem and current best solution
2: population $\leftarrow$ `initializePopulation`(P_0)
3: **if** $s_r \neq$ null **then**
4: population $\leftarrow$ `replace`(population, s_r)
5: **end if**
6: **while** termination criterion is not met **and not** proof **do**
7: $(\text{parent}_1, \text{parent}_2) \leftarrow$ `select`(population)
8: child $\leftarrow$ `crossover`($\text{parent}_1, \text{parent}_2$)
9: $(\text{child}, c_c) \leftarrow$ `tabuSearch`(child)
10: population $\leftarrow$ `replace`(population, child)
11: **if** $c_c < c_{\text{best}}$ **then**
12: $c_{\text{best}} \leftarrow c_c$
13: `send`(server, child) // send better solution
14: **end if**
15: $(s_r, \text{proof}) \leftarrow$ `receive`(server) // try to receive useful information
16: **if not** proof **then**
17: **if** $s_r \neq$ null **then**
18: population $\leftarrow$ `replace`(population, s_r)
19: **end if**
20: **end if**
21: **end while**

(line 16) CBC tries to obtain any published information from the server. If applicable (line 17), the best solution, cost and lower bound are updated (line 18). Better feasible solutions and costs are always useful for the BC algorithm because they provide efficient cutting planes. If the LP relaxation cost c_P of some active problem in L is even greater than the received better cost c_r (better upper bound), these active problems are removed from the list by function `filter`. As a result, the search space can be extremely reduced in some cases. Finally (line 27), any better lower bound can also be systematically sent to the server, which can be useful, especially when other algorithms have found a solution with the same cost, without optimality proof though.

5. Results

The methods described in the previous section have been implemented and thoroughly tested on a set of realistic en-route traffic instances. This section details the construction of these instances, which have been made publicly available, and presents the results obtained for conflict resolution.

5.1. Benchmark

To assess the performance of our resolution methods, we tested them on custom traffic scenarios within a fixed airspace volume. Figure 13 describes the construction of the scenarios. The initial position of each aircraft (the points marked O_k in the figure) is

Algorithm 4 Collaborative Branch and Cut (CBC)

1: $(P_0, s_r, c_s, lb_s) \leftarrow$ `register`(server) // get useful information
2: $(s_{P_0}, c_{P_0}) \leftarrow$ `LPrelax`(P_0)
3: $L \leftarrow [(P_0, s_{P_0}, c_{P_0})]$
4: **if** $s_r \neq$ null **or** $lb \neq$ null **then**
5: $\quad s_{best} \leftarrow s_r$, $c_{best} \leftarrow c_s$, $lb \leftarrow lb_s$
6: **else**
7: $\quad s_{best} \leftarrow$ null, $c_{best} \leftarrow +\infty$, $lb \leftarrow c_{P_0}$
8: **end if**
9: **while** $L \neq \varnothing$ **and** $c_{best} - lb > \epsilon$ **do**
10: $\quad (P, s_P, c_P) \leftarrow$ `pop`(L)
11: $\quad (P_c, s_c, c_c) \leftarrow$ `searchCut`(P, s_P, c_P)
12: $\quad$ **if** $s_c \neq$ null **and** $c_c < c_{best}$ **then**
13: $\quad\quad$ **if** s_c is integral **then**
14: $\quad\quad\quad s_{best} \leftarrow s_c, c_{best} \leftarrow c_c$
15: $\quad\quad\quad$ `send`(server, s_{best}) // send better solution
16: $\quad\quad\quad (s_r, c_r, lb_r) \leftarrow$ `receive`(server) // try to receive useful information
17: $\quad\quad\quad$ **if** $s_r \neq$ *null* **or** $lb_r \neq$ *null* **then**
18: $\quad\quad\quad\quad s_{best} \leftarrow s_r, c_{best} \leftarrow c_r, lb \leftarrow lb_r$ // update best solution, cost and bound
19: $\quad\quad\quad\quad L \leftarrow$ `filter`(L, c_r)
20: $\quad\quad\quad$ **end if**
21: $\quad\quad$ **else**
22: $\quad\quad\quad (P_1, P_2) \leftarrow$ `branch`(P_c)
23: $\quad\quad\quad (s_{P_1}, c_{P_1}) \leftarrow$ `LPrelax`(P_1)
24: $\quad\quad\quad (s_{P_2}, c_{P_2}) \leftarrow$ `LPrelax`(P_2)
25: $\quad\quad\quad$ `add`$(L, (P_1, s_{P_1}, c_{P_1}), (P_2, s_{P_2}, c_{P_2}))$
26: $\quad\quad\quad lb \leftarrow \min_{\forall P \in L} c_P$
27: $\quad\quad\quad$ `send`(server, lb) // send better lower bound
28: $\quad\quad$ **end if**
29: $\quad$ **end if**
30: **end while**

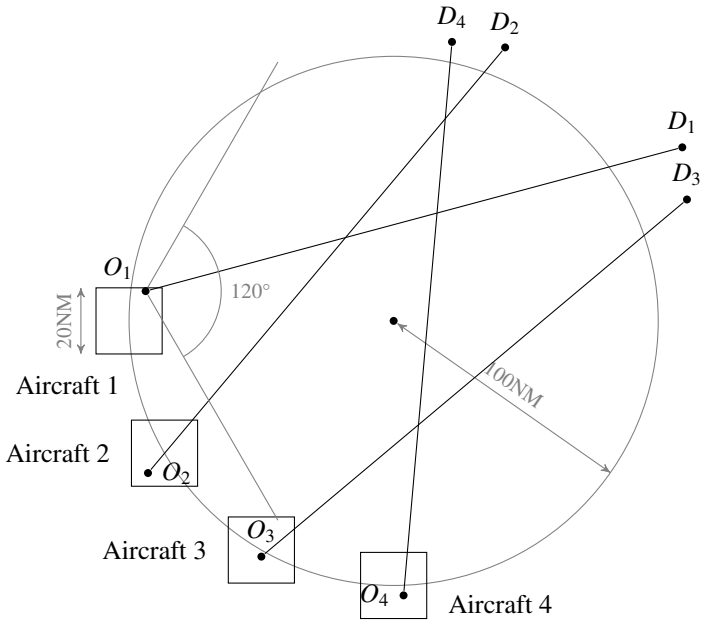


Figure 13: Geometry of conflict scenario generation.

randomly chosen within a 20 NM-wide square whose center lies on a 100 NM-radius circle. The route for each aircraft (which corresponds to the initial heading in these scenarios) is also randomly chosen in a ±60° interval around the heading that would lead the aircraft towards the center. Aircraft are equally (and randomly) dispatched among five FLs, ranging from FL280 to FL320. Finally, the nominal speed for each aircraft is randomly chosen in a ±20 % interval around 480 kn, a typical speed for airliners. The nominal vertical speed for maneuvers is set at 600 ft min^{-1} for all aircraft. The density of scenarios is controlled by the number of aircraft in the airspace volume. For the experiments reported in the current study, this number varies from 15 to 60. Based on this initial state, all possible trajectories are computed for every aircraft, using the maneuver parameters previously described in Table 1, then conflicts are detected with a medium uncertainty level corresponding to $E_{t_0} = 20$ s, $E_{t_1} = 20$ s, $E_\alpha = 2°$, $E_{v_h} = 4$ % and $E_{v_v} = 10$ % (cf. Table 2), using the method described in Section 3.2.4, resulting in a conflict matrix. As each scenario is randomly generated, we produced 10 different instances for each density to avoid the bias of some instances being particularly easy or difficult to solve. All the generated instances are available to download at `clusters.recherche.enac.fr`.

Real upper airspace sectors are generally smaller than the airspace volume we consider (up to 70 NM wide in France), but can be merged together when the traffic is not dense. Vertically, five FLs is a lower bound of what can be found in the real upper airspace. With the tools currently available, an air traffic controller can hardly handle more than 30 aircraft at a time in a 70 NM sector. Also, aircraft flying in the upper airspace prefer flying a direct routes at optimal FL, while the similarity of current

airliners performances tend to concentrate the demand on a small set of FLs. Thus, automatic solvers should favour bigger airspace and may have to deal with high demand on a small amount of FLs, hence the design of our scenarios.

5.2. Experimental Setup

We report in this section the performances of the various techniques described in Section 4 to solve the instances of the benchmark described in Section 5.1. All experiments were carried out on a standard workstation consisting of a 3.4 GHz Intel® Xeon® octo-core processor with 16 GB of memory running Debian GNU/Linux 9.4. We used the Gurobi Commercial Optimizer [20] for the ILP model and the ZeroMQ messaging library [2] to implement our cooperation framework.

As the MA is a stochastic algorithm using a pseudo-random number generator, 20 runs with different seeds were attempted for each instance to benefit from the diversification of the algorithm, and the best one is reported. All tests were done with a population of 50 individuals with 1000 iterations for the tabu search phase.

In the following sections, we first compare our previous published Constraint Programming (CP) approach [4] to the ILP model solved with Gurobi to show the limits of CP with such large-scale instances. Then, we show that the Memetic Algorithm and the ILP solver both easily reach optimal solution to small instances of the problem. We discuss their strengths and weaknesses on larger instances, then detail how their cooperation described in Section 4.3 largely outperforms any single algorithm and scales well with larger and more complex instances in a limited amount of time (300 s).

5.3. Analysis

As mentioned in 5.1, the limited number of FLs and high density of the generated instances make the conflict resolution quite challenging. Moreover, whereas ATC usually tries to find a feasible maneuver for aircraft involved in a conflict, our approach also focuses on the cost of the solution in terms of fuel consumption, aiming at an optimum of the cost function described in Section 3.

Figure 14 shows the mean time (in seconds) needed to compute the optimal solution for both ILP and CP algorithm for all instances from 15 to 30 aircraft. The ILP solver clearly outperforms the CP one, up to several orders of magnitude for the largest instances with 30 aircraft. If we considered 300 s limited time to run each algorithm, to obtain a first solution with CP can be challenging or even out of reach within the allocated time. So in the following, we only present performances of MA and ILP algorithm, especially their cooperation. Note that execution times in Figure 14 are plotted with a base-10 logarithmic scale.

For small 3D instances up to 40 aircraft, both MA and ILP algorithm obtain an optimal solution, and the optimality can always be proved by ILP in very limited time. Figure 15 shows the mean time needed to compute the optimal solution for both MA and ILP for all instances ranging from 20 to 40 aircraft. Using ILP, the optimum is found within a few seconds.

For larger instances however, the problem is more challenging, and finding the optimal might be over-costly. In order to stay within the limits of a potential real-time application, we decided to limit the computation time to 300 s. Figure 16 shows the

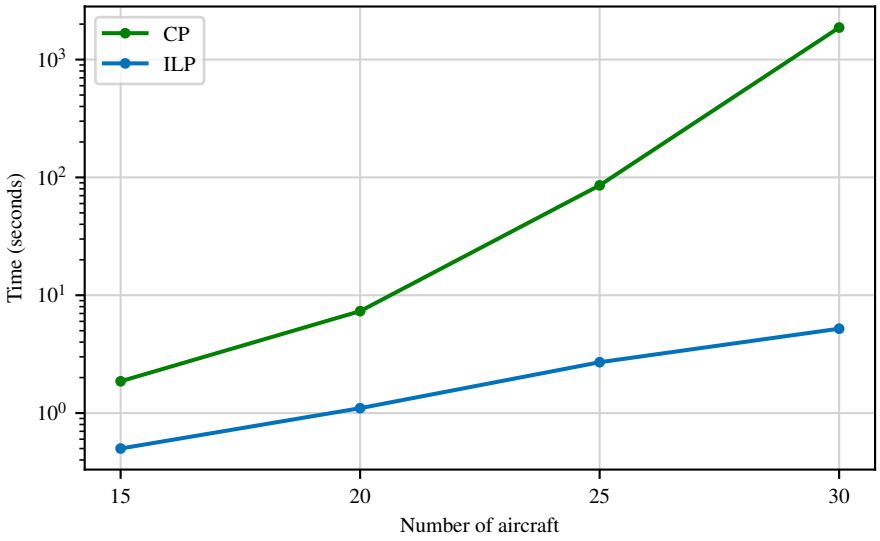


Figure 14: Comparison of computation times (in logarithmic scale) to find an optimal solution for small instances with CP and ILP.

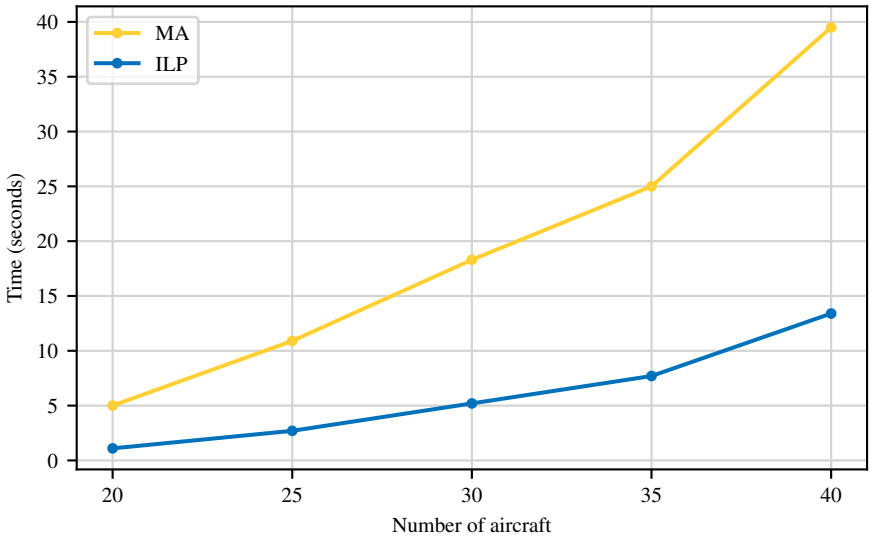


Figure 15: Comparison of computation times to find an optimal solution for small instances with MA and ILP.

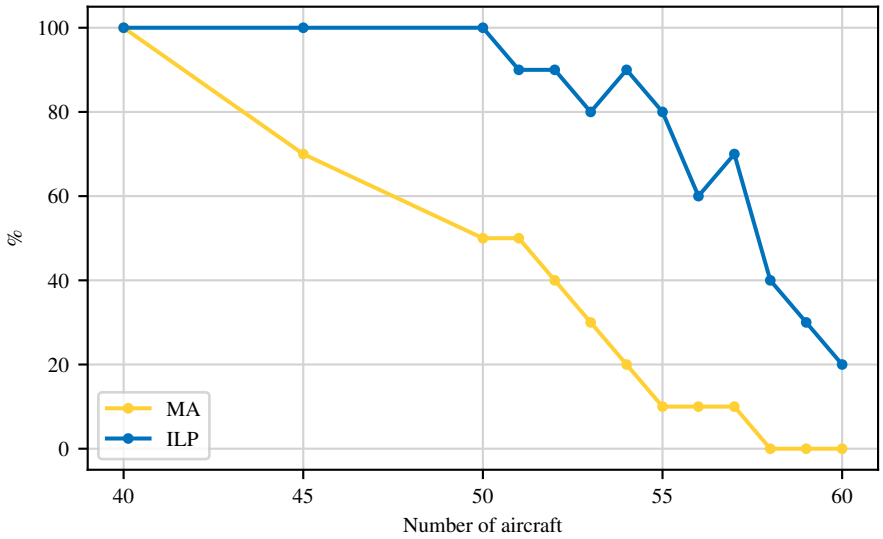


Figure 16: Percentage of success for finding an optimal solution within a 300 s time limit with MA and ILP.

percentages of large instances (ranging from 40 to 60 aircraft) for which an optimal solution was found. This ratio quickly decreases when the density increases, with ILP still being able to optimally solve about 20 % of our largest instances.

Next section shows that the cooperation between MA and ILP significantly increases the success rate of finding an optimum, and thus enhances the cost of the resulting solution.

5.4. Cooperation

We present in this section the results obtained with the cooperative version of the MA (cf. Algorithm 3) and of the Branch and Cut (cf. Algorithm 4) integrated in the distributed framework described in Section 4.3. The behavior of the resulting solver exhibits a much better behavior when the density of instances increases. It gives optimal solutions within the 300 s time limit for almost all feasible instances, while MA or ILP alone could not reach the optimum, as depicted in figure 17.

In figure 18, we compare the cost (averaged over 10 runs for each density) of the best solutions found by the MA, ILP and their cooperation, with respect to the optimal value. Here, 100 % represents the optimum cost, while a larger value represents the cost of a non-optimal solution. As expected from our previous observations, the cooperation systematically reaches an optimal solution. For even larger instances (with 70 to 100 aircraft), MA or ILP alone can be much farther from the optimal value: up to 3.5 % on average, and up to 10 % on some particular instances.

Figure 19 shows the evolution of the cost of the best solution during the search for one of the most difficult instances in our set, involving 59 aircraft. First, we observe that, at the end of the 300 s limit, the cost of the solutions found by MA and ILP are similar, but are about 10 % higher than the cost of the cooperation solution. Second, we

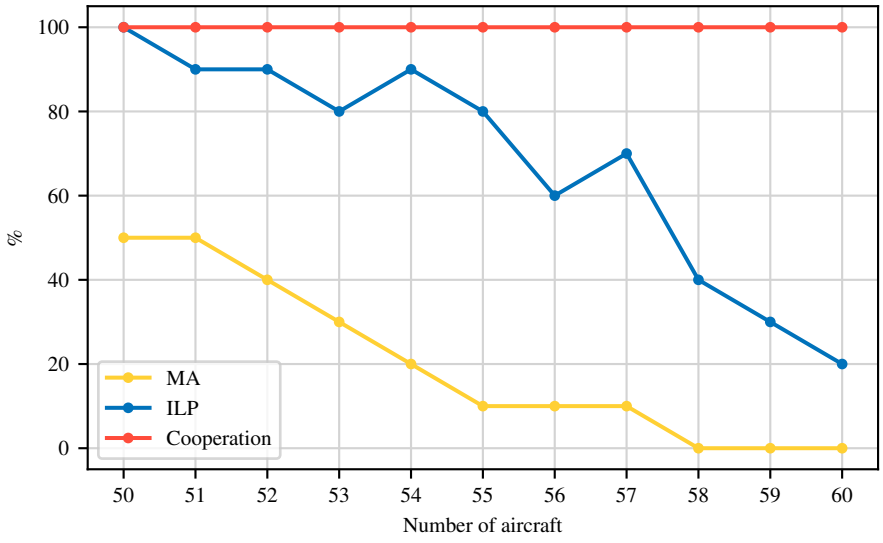


Figure 17: Percentage of success for finding an optimal solution with a 300 s time limit with MA, ILP and their cooperation.

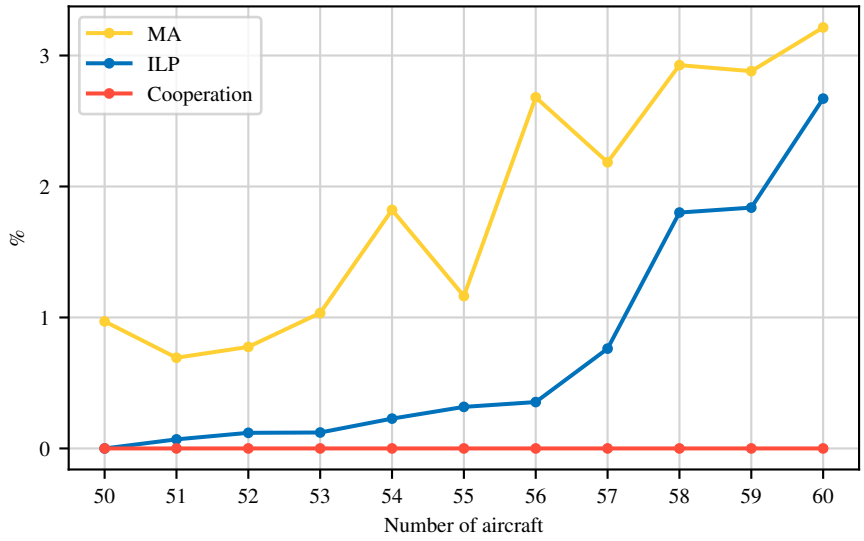


Figure 18: Average cost of the best solution found within 300 s, expressed as the ratio of the difference to the optimum.

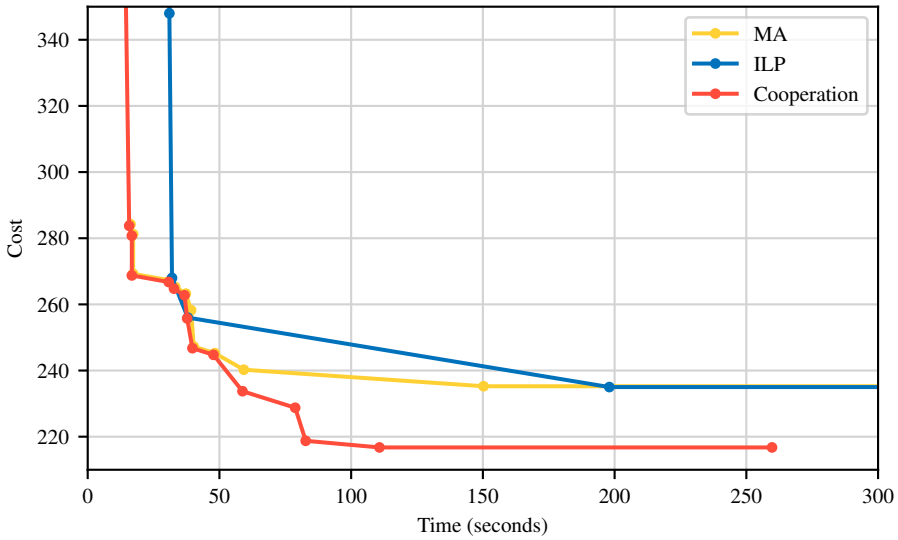


Figure 19: Convergence of the cost with a 300 s time limit.

can see from the graph that the cooperation proved the optimality of its solution, as the process stopped at about 260 s. This optimal solution was found after about 110 s. In the meantime, MA and ILP seemed to be stuck, probably on a locally optimal solution, after 150 s to 200 s. In the first part of the search, the cooperation follows exactly the same convergence profile than the MA, while the ILP algorithm does not provide any solution. This is due to the fact that the solver used for ILP performs some internal transformations of the problem to provide a more efficient search afterwards. During this process, no solution can be proposed. The cooperation highly benefits from this mechanism, as in the meantime the MA can quickly converge to solutions with a good cost, thus saving time for the ILP solver when it is ready to execute the BC algorithm.

In order to assess how the cooperation impacts the optimality of the solutions, we performed a new series of tests without the 300 s time limit, and measured the time needed for the ILP solver alone and for the cooperation to find and prove the optimal solution on large instances (50 to 60 aircraft). The results of this experiment are shown on figures 20 and 21 respectively. Note that the y-axis is in log scale on both figures.

Figure 20 shows that the advantage of the cooperation over the ILP solver alone increases with the density of the problem: for instances around 50 aircraft, it is only 1.2 to 2 times faster to find the optimum, while for larger instances, it can be up to 10 times faster.

For the proof of optimality, we see on figure 21 that the difference is more consistent, with the cooperation being approximately 2 times better than ILP alone. This is explained by the fact that the MA does not contribute to the proof of optimality due to its lack of completeness properties. Thus, once an optimal solution has been found, the cooperation has no further advantage over the ILP solver alone. Also, we observe that proving optimality is significantly costlier than finding an optimal solution (by a

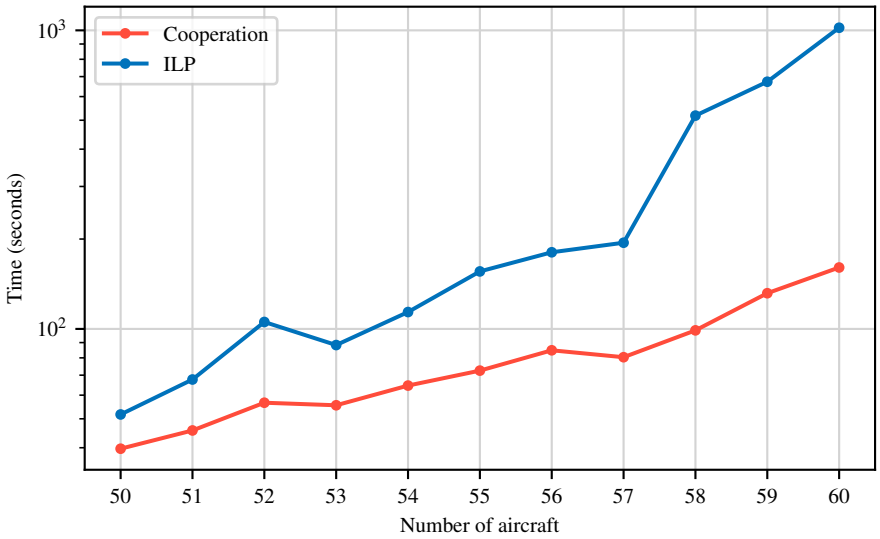


Figure 20: Comparison of computation times to find an optimal solution for the cooperation and the ILP solver alone.

10 to 100 factor). In the reported experiment, it was even out of reach (in a reasonable computation time) for most 60-aircraft instances.

5.5. Infeasible Instances

In an operational context, any real-time software that manages a non-interruptible critical system should provide a fallback scheme, were an instance infeasible, i.e. without valid conflict-free solutions. In such cases, the solver should provide a set of maneuvers "as good as possible", trying to minimize the number and severity of conflicts, and report perilous situations.

Table 3: Value of the uncertainty bounds w.r.t. the error level l.

uncertainty bound	E_{t_0}	E_{t_1}	E_α	E_{v_h}	E_{v_v}
value for level l	$l \times 10\,\mathrm{s}$	$l \times 10\,\mathrm{s}$	$l \times 1°$	$l \times 2\,\%$	$l \times 5\,\%$

One of the main advantages of combining a complete algorithms (like Branch and Cut) able to prove optimality or infeasibility with a metaheuristic (like an MA) is that the latter directly processes candidate solutions in the maneuver space, possibly with remaining conflicts. Moreover, the minimization of the number of conflicts is the main criterion used by its objective function, before attempting to minimize its maneuver cost as described in Section 4.1.2. Consequently, when no legal solution is found, the best current solution of the MA tends to minimize the number of conflict violations.

Figure 22 shows the mean of the number of remaining conflicts over five scenarios with 60 aircraft, w.r.t. the level of uncertainty $l \in \{5, \ldots, 9\}$ of the detection phase, which

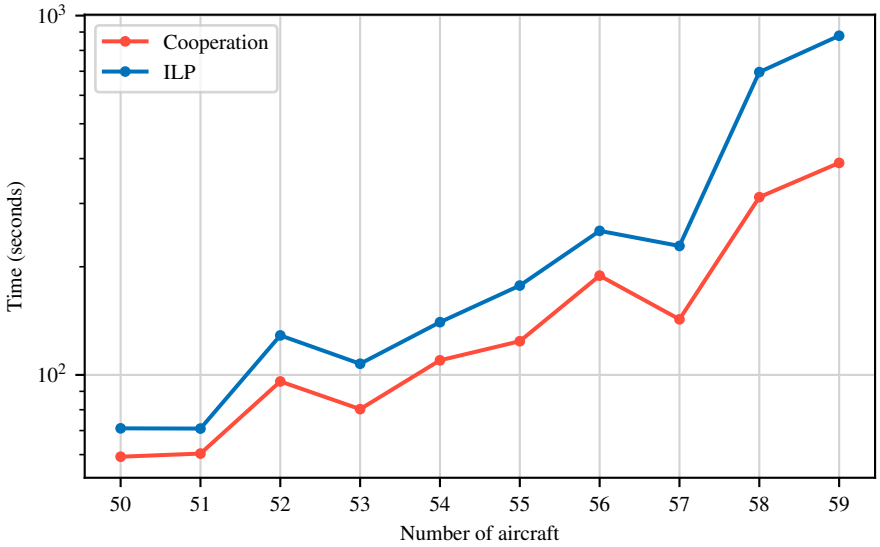


Figure 21: Comparison of computation times to prove optimality for the cooperation and the ILP solver.

was artificially raised until the problem becomes unfeasible (e.g. for $l = 9$, the aircraft may deviate up to 9° relatively to the requested heading change, which is much more than operational error levels). Indeed, envelopes representing the position of aircraft will expand with the level of uncertainty, as well as the number of conflicts. Table 3 gives the value of the various uncertainty bounds defined in Section 3.2.2 w.r.t. to the uncertainty level l. Note that the ILP solver is consistently able to report the infeasibility of the tested instances.

In such a case, our cooperation framework can be adapted in several ways:

- whenever a complete solver proves that the instance is infeasible, the mode of the cooperation can be switched to report the best solution obtained by a metaheuristic within the time limit;
- similarly, if no valid solution nor proof of infeasibility were obtained within the time limit, the best solution of the MA can be reported as well;
- eventually, provided that there is enough time to restart the search, the separation norm could be reduced until the problem becomes solvable.

Moreover, in an operational context, all conflicts are not equivalent as violations can be almost inconsequential (e.g. the closest point of approach is 4.5 NM instead of the required 5 NM in the horizontal plane) or severe (e.g. less than 0.5 NM), only last a fraction of a second or continue during one minute. To handle infeasibility more precisely, we could compute the severity of each potential conflict during the detection phase described in Section 3.2.4 and store the information in the conflict matrix. The objective function of the MA can then be modified to minimize the sum of conflict violations weighted by their severity to obtain the safest maneuver set.

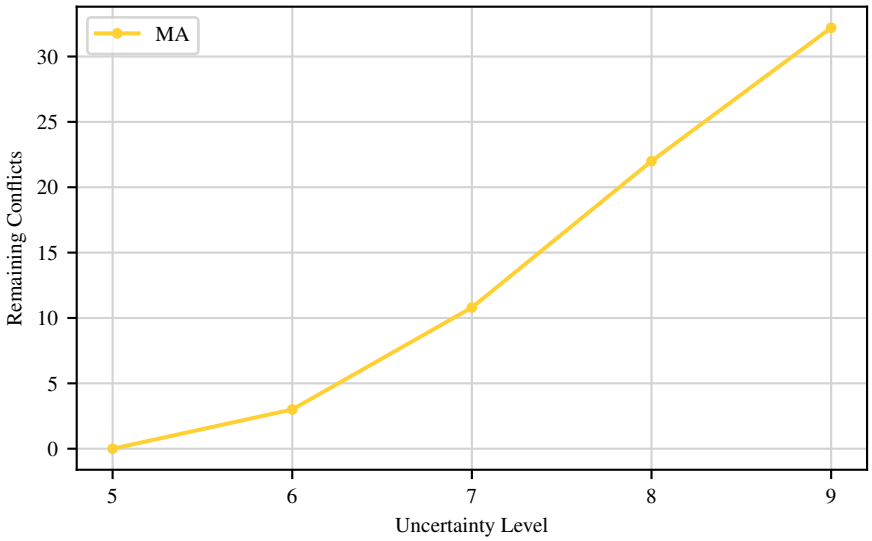


Figure 22: Average of the number of remaining conflicts over 5 instances with 60 aircraft w.r.t. the level of uncertainty l, resolved by the MA with a 300 s time limit.

Furthermore, in an operational setting, conflict resolution would be iterated over a Rolling Horizon taking into account a limited time window (e.g. 20 min), which is then shifted by a given time step (e.g. 5 min). Therefore, the volume of a given aircraft position envelope (cf. Section 3.2.3) at a given time in the current window will shrink in the next one, as the envelope will be closer to the initial position of the aircraft. Consequently, conflicts that cannot be solved in the current time window might be easier to solve during a later iteration, or even disappear. As explained in Section 4.1.2 concerning the cost of valid solutions, the MA could also be tuned to favor solutions with remaining conflicts appearing as late as possible, in the hope that the reduction of uncertainty will make them easier to solve during the next iterations.

Eventually, the range of maneuvers could also be extended to widen the solution space with, for example, heading changes of 5, 15 and 25°, vertical changes of ±3000 ft or speed adjustments greater than a −6 % slow down or greater than a 3 % speed up.

6. Conclusion and Further Work

We have presented an innovative framework for the modeling and resolution of en-route conflicts in three dimensions. The modeling is clearly separated from the resolution, thus giving the opportunity to compare various resolution methods on the same instances. Horizontal, vertical and speed maneuvers are taken into account, and a comprehensive uncertainty model is described as well. The proposed modeling is also totally generic, and gives users the possibility to integrate their own maneuvers. Based on our model, a large set of realistic instances of the problem have been generated, with

various densities and difficulties. These instances have been made freely available to the research community at `clusters.recherche.enac.fr`.

Building on previous work (see [3]), we have proposed two algorithms for the resolution of the en-route conflict problem: a Memetic Algorithm (MA) and a Branch and Cut (BC) to solve Integer Linear Programs (ILPs). The MA has the benefit of finding feasible solutions in a very short time, even for dense instances, while the ILP was able to find and prove optimal solutions for low to medium density instances. Based on these observations, we have designed a generic framework for the cooperation of optimization solvers, in which any algorithm can share information such as partial solutions, lower or upper bounds for the cost, etc. This framework has been successfully tested with the MA and ILP solver.

Instances of low density were optimally solved by all three methods within less than one minute. For larger instances, we have restricted the computation time by 300 s, in order to make the approach compatible with a real-time context. With this limit, both the MA and ILP solver were able to provide good solutions to high density instances, without reaching optimality though. The cooperation between the MA and the ILP solver outperformed both approaches on all instances, and made it possible to reach and prove optimality in most cases, even for very dense instances. Moreover, should a particular instance be infeasible, our solver is able to provide a set of maneuvers that minimizes the number of remaining conflicts, thanks to the incremental properties of the MA.

The proposed framework is completely generic, from the model to the cooperation method, which provides many opportunities for future research. On the optimization side, we could plug other algorithms in the cooperation tool to further enhance the performance of the resolution. We could also weigh the conflicts according to their severity during the detection phase to handle infeasibility with more accuracy and provide "as-good-as-possible solution". Regarding the Technology Readiness Level of our solver, our next step is to integrate this framework into a more realistic air traffic simulator, with a finer model of aircraft performances, to validate the approach before assessing its real-life abilities in an operational context.

References

[1] Achterberg, T., Bixby, R. E., Gu, Z., Rothberg, E. and Weninger, D. [2014], Multi-row presolve reductions in mixed integer programming, *in* 'Proceedings of the Twenty-Sixth RAMP Symposium', Tokyo.

[2] Akgul, F. [2013], *ZeroMQ*, Packt Publishing.

[3] Allignol, C., Barnier, N., Durand, N. and Alliot, J.-M. [2013], A new framework for solving en-routes conflicts, *in* '10th USA/Europe Air Traffic Management Research and Developpment Seminar'.

[4] Allignol, C., Barnier, N., Durand, N., Gondran, A. and Wang, R. [2017], Large scale 3D en-route conflict resolution, *in* 'ATM Seminar, 12th USA/Europe Air Traffic Management R&D Seminar'.

[5] Alonso-Ayuso, A., Escudero, L. F. and Martín-Campo, F. J. [2011], 'Collision avoidance in air traffic management: a mixed-integer linear optimization approach', *IEEE Transactions on Intelligent Transportation Systems* **12**(1), 47–57.

[6] Alonso-Ayuso, A., Escudero, L. F. and Martín-Campo, F. J. [2016*a*], 'An exact multi-objective mixed integer nonlinear optimization approach for aircraft conflict resolution', *Top* **24**(2), 381–408.

[7] Alonso-Ayuso, A., Escudero, L. F. and Martín-Campo, F. J. [2016*b*], 'Multiobjective optimization for aircraft conflict resolution. a metaheuristic approach', *European Journal of Operational Research* **248**(2), 691–702.

[8] Averty, P., Johansson, B., Wise, J. and Capsie, C. [2007], Could erasmus speed adjustments be identifiable by air traffic controllers, *in* '7th USA/Europe air traffic management research and development seminar (ATM2007)', Vol. 22.

[9] Awad, N. H., Ali, M. Z., Suganthan, P. N. and Reynolds, R. G. [2017], 'CADE: a hybridization of cultural algorithm and differential evolution for numerical optimization', *Information Sciences* **378**, 215–241.

[10] Barnier, N. and Allignol, C. [2012], 'Trajectory deconfliction with constraint programming', *The Knowledge Engineering Review* **27**(03), 291–307.

[11] Blum, C., Cotta, C., Fernández, A. J., Gallardo, J. E. and Mastrolilli, M. [2008], Hybridizations of metaheuristics with branch & bound derivates, *in* 'Hybrid Metaheuristics', Springer, pp. 85–116.

[12] Blum, C., Puchinger, J., Raidl, G. R. and Roli, A. [2011], 'Hybrid metaheuristics in combinatorial optimization: A survey', *Applied Soft Computing* **11**(6), 4135–4151.

[13] Durand, N., Alliot, J.-M. and Noailles, J. [1996], Automatic aircraft conflict resolution using genetic algorithms, *in* 'Proceedings of the Symposium on Applied Computing, Philadelphia', ACM.

[14] Durand, N. and Granger, G. [2003], A traffic complexity approach through cluster analysis, *in* '5th ATM R&D Seminar'.

[15] El-Abd, M. and Kamel, M. [2005], A taxonomy of cooperative search algorithms, *in* 'International Workshop on Hybrid Metaheuristics', Springer, pp. 32–41.

[16] Erzberger, H., Paielli, R. A., Isaacson, D. R. and Eshowl, M. M. [1997], Conflict probing and resolution in the presence of errors, *in* 'Procceedings of the 1st USA/Europe Seminar'.

[17] Gilbert, E. G., Johnson, D. W. and Keerthi, S. S. [1988], 'A fast procedure for computing the distance between complex objects in three-dimensional space', *IEEE Journal on Robotics and Automation* **4**(2), 193–203.

[18] Graham, R. L. [1992], An efficient algorithm for determining the convex hull of a finite planar set, *in* 'Information Processing Letters'.

[19] Granger, G., Durand, N. and Alliot, J.-M. [2001], Optimal resolution of en-route conflicts, *in* '4th ATM R&D Seminar'.

[20] Gurobi Optimization, L. [2018], 'Gurobi optimizer reference manual'.
URL: *http://www.gurobi.com*

[21] Hao, J.-K. [2012], *Memetic Algorithms in Discrete Optimization*, Springer Berlin Heidelberg, Berlin, Heidelberg, p. 73–94.
URL: *http://dx.doi.org/10.1007/978-3-642-23247-3˙6*

[22] Heidari, A. A., Aljarah, I., Faris, H., Chen, H., Luo, J. and Mirjalili, S. [2019], 'An enhanced associative learning-based exploratory whale optimizer for global optimization', *Neural Computing and Applications* pp. 1–27.

[23] Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G. and Wolsey, L. A. [2009], *50 Years of integer programming 1958-2008: From the early years to the state-of-the-art*, Springer Science & Business Media.

[24] Lehouiller, T., Omer, J., Soumis, F. and Desaulniers, G. [2015], A flexible framework for solving the air conflict detection and resolution problem using maximum clique in a graph, *in* '11th USA/Europe Air Traffic Management Research and Developpment Seminar'.

[25] Lehouillier, T., Omer, J., Soumis, F. and Desaulniers, G. [2017], 'Two decomposition algorithms for solving a minimum weight maximum clique model for the air conflict resolution problem', *European Journal of Operational Research* **256**(3), 696–712.

[26] Li, R., Hu, S., Wang, Y. and Yin, M. [2017], 'A local search algorithm with tabu strategy and perturbation mechanism for generalized vertex cover problem', *Neural Computing and Applications* **28**(7), 1775–1785.

[27] Mirjalili, S. and Lewis, A. [2016], 'The whale optimization algorithm', *Advances in engineering software* **95**, 51–67.

[28] Mitchell, J. E. [2002], 'Branch-and-cut algorithms for combinatorial optimization problems', *Handbook of applied optimization* pp. 65–77.

[29] Pallottino, L., Féron, E. and Bicchi, A. [2002], 'Conflict resolution problems for air traffic management systems solved with mixed integer programming', *IEEE Transactions on Intelligent Transportation Systems* **3**(1), 3–11.

[30] Raidl, G. R. and Puchinger, J. [2008], Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization, *in* 'Hybrid metaheuristics', Springer, pp. 31–62.

[31] Rey, D. and Hijazi, H. [2017], Complex number formulation and convex relaxations for aircraft conflict resolution, *in* '2017 IEEE 56th Annual Conference on Decision and Control (CDC)', IEEE, pp. 88–93.

[32] Rey, D., Rapine, C., Fondacci, R. and El Faouzi, N.-E. [2012], 'Minimization of potential air conflicts through speed regulation', *Transportation Research Record: Journal of the Transportation Research Board* **2300**, 59–67.

[33] Richard, A., Gregory, A. and Nicolas, D. [2017], Efficient conflict detection for conflict resolution, *in* 'Submitted at the 12th USA/Europe Air Traffic Management Research and Developpment Seminar'.

[34] van den Bergen, G. [1999], 'A fast and robust GJK implementation for collision detection of convex objects', *Journal of graphics tools* **4**(2), 7–25.

[35] Vela, A., Solak, S., Singhose, W. and Clarke, J.-P. [2009], A mixed integer program for flight-level assignment and speed control for conflict resolution, *in* 'Proceedings of the Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference', IEEE.